Interoperability of Peer-To-Peer File Sharing Protocols

Siu Man Lui and Sai Ho Kwok

Peer-to-Peer (P2P) file sharing software has brought a hot discussion on P2P file sharing among all businesses. Freenet, Gnutella, and Napster are the three most popular P2P file sharing applications. They use three distinct protocols and these protocols come with different characteristics. In this paper, we discuss the protocols of these P2P file sharing applications, in terms of the methodologies used for peer registry, query and content sharing. In order to maximize the benefit of P2P file sharing application that is to facilitate file sharing among various P2P file sharing applications, we propose a framework to integrate various P2P file sharing protocols using P2P gateway in this paper.

Categories and Subject Descriptors: D.2.11 [**Software**] : Software Engineering – *Software Architecture,* D.2.12 [**Software**] : Software Engineering – *Interoperability*
General Terms: File sharing, file searching, peer
Additional Key Words and Phrases: Gnutella , Napster, Peer-to-Peer

1. INTRODUCTION

News headlines surrounding the lawsuit of RIAA against the Peer-to-Peer (P2P) file sharing software company, Napster, has brought a hot discussion of P2P file sharing among not only the music industry, but also other businesses. At the same time, P2P file sharing has gained a large acceptance among the internet users. A study found that over 1 million different peers have connected to the network in an 8-day period [11]. P2P computing is defined as the sharing of computer resources and services by direct exchange between systems. The resources and services include the exchange of information and content files, processing cycles, cache storage, and disk storage for data files. P2P takes advantage of existing desktop computing power and networking connectivity allowing economic clients to leverage their collective powers and benefit the entire community [10]. P2P networks replace the traditional client-server interactions with peer interactions, where each machine acts as both a client and a server in the network. Before the emergence of P2P file sharing application, the most common model for file sharing over the Internet was the client/server model, where there is a client, a browser or an ftp client that downloads the files from the server, which can be a web server or an ftp server. In this model, the communication is always initiated by the client request for the file and the server responds to the request by sending the requested file. In the P2P model, all peers in the network can both act as a client and a server at the same time. A peer can request files from others and share files with other peers.

Addresses: Department of Information and Systems Management, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China.
Email: {imcarrie, jkwok}@ust.hk

1. What are the differences and selection criteria for different P2P system architectures?
2. What are the problems of interoperability of these P2P file sharing protocols?
3. How can these protocols work together?

The structure of this paper will be as follows. Section 2 will be a discussion of the pros and cons of different P2P system architectures, namely, 1) pure P2P, and 2) server-mediated P2P. Then, section 3 will discuss the protocols of P2P file sharing application, namely Freenet [1], Gnutella [3], and Napster [6], in terms of the methodologies used for peer registry, query and content sharing. In this study, we focus the discussion on P2P file sharing protocols, therefore, the Internet's Service Location Protocol, the JINI Advertisement of Service Model, ebXML client, and SAML Authorities are not considered. Finally, we will study the interoperability of the file sharing protocols and issues involved for the integration, then we conclude the paper in section 5.

2. P2P FILE SHARING ARCHITECTURE

P2P file sharing application can be classified into two distinct classes according to their architectures. One is the "pure" P2P architecture as shown in Figure 1, which consists of peers that possess identical capabilities. Besides, there is no central server in this architecture. Another one is "server-mediated" P2P architecture as depicted in Figure 2. In this architecture, each machine has its distinct role. The central server is responsible to maintain a registry of shared information and respond to queries for that information. The peers are responsible to host the information, declare what information is shared to the central server, and download information from other peers upon requests.

2.1 Pure P2P

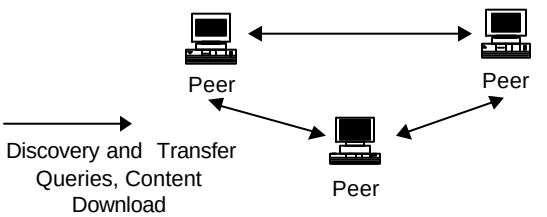

Figure 1. An example of a pure p2p architecture.

A pure P2P architecture has no central server as shown in Figure 1. It dynamically discovers other peers on the network and interacts with each of them by sending and receiving digital messages. The strength of pure P2P architecture is that it does not rely on the availability of any particular server for network location registration in order for other peers to find it. However, this architecture poses a problem that a relatively less number of clients can be discovered and thereby restricting the application's reach. In this scenario, a peer can either use information from a local configuration scheme to discover clients (for example, a configuration entry that tells it who to talk with) or a peer can employ network broadcasting and discovery techniques such as IP multicast to discover other peers. Using IP multicast can be problematic since it is not widely deployed to the Internet. However, it can be useful in Intranet scenarios where the network is more controllable and the infrastructure required for IP multicast is available for use. Multicast is also problematical because it invites denial of service attack, often originating in software bugs rather than intentional attacks, and is therefore

forbidden in many locations. Pure P2P is also deployed to the Internet in the case that non-multicast schemes are used to discover peers. In this case, the application may use some other schemes for peer discovery such as a well-known approach that each peer knows about at least one other peer and they share this knowledge with other peers to form a loosely connected mass of peers.

2.2 Server-Mediated P2P

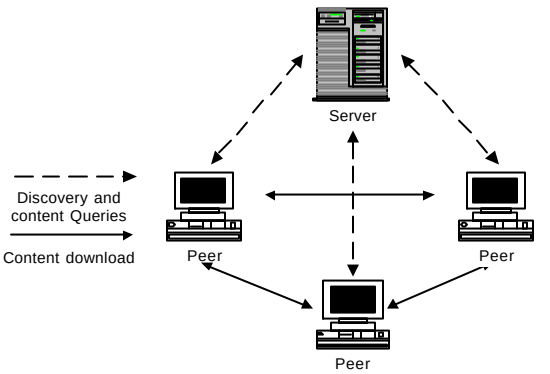

Figure 2. An example of server-mediated p2p architecture.

The server-mediated P2P architecture works just like the pure P2P architecture except that it relies on a central server for peer discovery and content lookup. In this model, the P2P file sharing application usually notifies the central server of its existence at startup time (or user login time). The application then uses this server to download a list of other peers participating on its network. When an application is looking for some particular contents, it queries the central server rather than sending queries to each peer. The central server then responds with a list of the peers that possess the requested content and the peer application can contact those peers directly to retrieve the content. In many case, it is easier to make this solution scale better than the pure P2P option because peer discovery and content lookup only need to send a message to the central server instead of all peers. It is noted that it is possible to make pure P2P solutions scale extremely well, but if you are able to rely on a server for some of the basic tasks (like discovery and content lookup), high scalability can be achieved at a lower cost in terms of development time. However, this approach hinges on the availability of the central server. If the central server is not available, the P2P application will not be able to find other peers.

Table 1. Pros and Cons of different P2P architectures.

P2P architecture	Pros	Cons	Examples
Pure P2P	- No central server - Higher fault tolerant - Simpler architecture	- More network resource consumption - Low scalability	- Gnutella, - Freenet
Server-mediated P2P	- Less network resource consumption - High scalability	- Central server dependent - Less fault tolerant	- Napster

In addition, requesting content from peers can be quite expensive from a network resource perspective. This may not seem a big deal if there are only a few peers

interacting over the network, but if your application is being written for use over the Internet or the cross-enterprise and cross-country environment, this consideration suddenly becomes much more significant scaling factorial. We summarize the pros and cons of the pure P2P and server-mediated P2P architecture in Table 1.

3. P2P PROTOCOLS

Freenet, Gnutella, and Napster are the three most popular P2P file sharing applications. They use three distinct protocols and these protocols come with different characteristics. Freenet and Gnutella use the pure P2P architecture while Napster has a central server to handle peer discovery and content lookup. In the following, we will illustrate their protocols in details.

3.1 Freenet

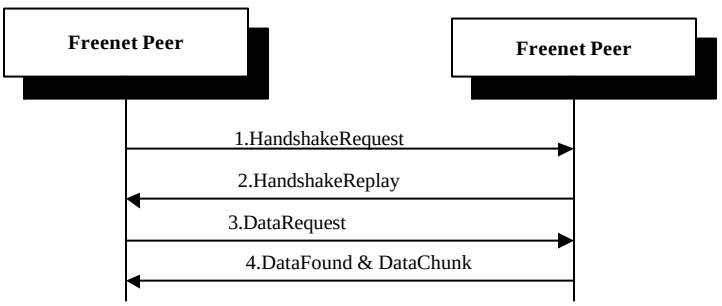

Figure 3. Logical flow of the Freenet messages.

Table 2. Summary of Freenet Protocol.

Message Type	Description
HandshakeRequest	This message is used to discovery other peer in the Freenet network.
HandshakeReplay	This message is a reply to the HandshakeRequest message, which specifying the protocol version number that the peer can understand.
DataRequest	This message specifies a search key, the remote node will check the datastore for the key.
DataFound	This message is used to indicate that the request data is matched in the peer.
DataChunk	This contains the actual data sent to the peers.

Freenet is a free software project, developed by Ian Clarke [5] at the University of Edinburgh. Freenet is implemented as an adaptive P2P network of peers that query one another to store and retrieve data files, which are named by location-independent keys. Each peer maintains its own local datastore and the datastore makes available to the network for reading and writing, as well as a dynamic routing table containing network addresses of other peers and the keys that they hold. All messages between the Freenet peers contain a randomly generated 64-bit transaction ID, a hops-to-live limit, and a depth counter. Hops-to-live is set by the originator of a message and is decremented at

each hop to prevent message being forwarded indefinitely. Depth is incremented at each hop and is used by a replying node to set hops-to-live high enough to research a requestor.

The peer in the Freenet network identifies each other by sending a "HandshakeRequest" message to another peer. If the remote peer is active and responding to the request, it will reply with a "HandshakeReply" message specifying the protocol version number that it understands. Handshakes are remembered for a few hours, and subsequent transactions between the same nodes during this time may omit these steps. To find a file in the Freenet network, the sending node sends a "DataRequest" message specifying a transaction ID, initial hops-to-live and depth and a search key. The remote node will check its datastore for the key and if not found, will forward the request to another node. Using the chosen hops-to-live limit, the sending node starts a timer for the expected amount of time it should take to contact that many nodes, after which it will assume failure. If the request is successful, the remote node will reply with a "DataFound" message containing the address of the node that supplied it and the length of the requested data. After the "DataFound" message, the data itself is sent in chunks with the "DataChunk" message. Figure 3 shows the logical flow of the Freenet messages between two Freenet peers, and Table 2 summarizes the messages in the Freenet protocol [2].

3.2 Gnutella

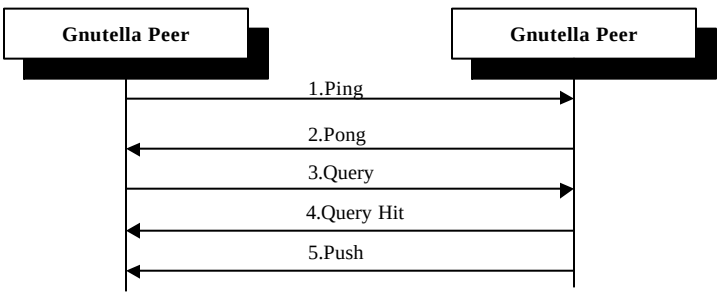

Figure 4. Logical flow of the Gnutella messages.

Gnutella represents a new wave of P2P applications providing distributed discovery and sharing of resources across the Internet. Gnutella is distinguished by its support for anonymity and by its decentralized architecture. A Gnutella network consists of a dynamically changing set of peers connected using TCP/IP. Each peer acts as a client (an originator of queries), a server (a provider of file information), and as a router (a transmitter of queries and responses). At any given point in time, the Gnutella network consists of a set of interconnected peers. The Gnutella protocol defines the way in which peers communicate over the network. It consists of a set of descriptors used for communicating data between peers and a set of rules governing the inter-servant exchange of descriptor. Figure 4 shows the logical flow of the Gnutella messages between two Gnutella peers. Currently the five types of messages are defined in the Gnutella protocol specification v0.4 [4] as shown in Table 3. Peers discover each other by sending the "Ping" and "Pong" messages. The file query and downloading for a Gnutella peer involves three steps. In the first step, a "Query" message is sent out. The massage contains a string specifying the set of files that the user request. Each peer that receives the message uses this string to determine which files, if any, match the query. In the second step, a peer that matches a query generates one or more "QueryHit" message

giving information about obtaining the file. In the third step, the query peer connects directly to the response peer and uses the simplified version of hypertext transfer protocol (HTTP) to retrieve the file using the returned uniform resource locator.

Table 3. Summary of the Gnutella protocol.

Message Type	Description
Ping	A peer sends a Ping message to discover hosts on the network. A peer receiving a Ping descriptor is expected to respond with one or more Pong message.
Pong	A peer sends a Pong message to response to a Ping message. The responding peer provides its IP address and number of sharable files.
Query	A peer sends a Query message to locate a set of files matching some filter criteria. A peer receiving a Query message will respond with a QueryHit if a match is found in its local data store.
QueryHit	A peer sends a QueryHit message to response to a query giving a list of files matching the filter criteria and the IP address of the provider. This descriptor provides the recipient with enough information to acquire the data, matching the corresponding Query.
Push	Push is a mechanism that allows a firewalled peer to contribute file-based data to the network.

3.3 Napster

Unlike Freenet and Gnutella, Napster network contains a central sever. Therefore, the Napster protocol is divided into peer to server communications and P2P communications. Each message is composed of three fields; they are “length”, “type” and “data”. The fields of “length” and “type” are 2 byte-long data. The “length” field specifies the length in bytes of the “data” portion of the message. The “type” field specifies the type of message and the “data” portion of the message is a plain ASCII string.

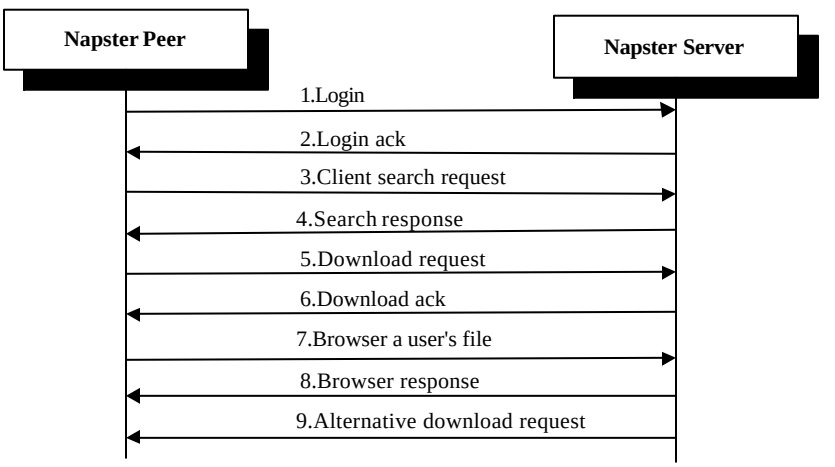

Figure 5. Logical flow of the Napster messages.

Instead of discovering other peers using “Ping” or “HandshakeRequest” used in Gnutella and Freenet, a Napster peer will login to a Napster server, which is responsible to maintain a peer list and content directory. In order to login to a Napster server, a Napster peer sends a “Login” message to the server by providing its nickname, password

and port number open for Napster file sharing. Then the server will reply with a "Login ack" message to indicate the successful login. For search and downloading file, the downloading peer will first issue either a "Peer search request" or "Browse a user's file" message to the server. The server responses with the "Search response" and the "Browse response" message respectively. These messages provide information such as the nickname of the user who contributes the file, the filename, the size of the file. To request a download, a "Download request" message is sent to the server.

Table 4. Summary of the Napster Protocol.

Message Type	Description
Login	Napster peer sends this message to the server. This contains the nickname and password for Napster login. It also provides the port number and the speed of the Internet connection of the client.
Login ack	Napster server sends this message to the peer to indicate a successful login.
Client search request	Napster peer sends this message to the server to specify the keyword for file searching and the maximum results for return.
Search response	Napster server will send this message to the peer when a match file is found. This contains the details about the matched files.
Download request	Napster peer sends this message to the server with the nickname of the peer, which hosts the file and the requested filename.
Download ack	Napster server sends this message to the peer with the details of the file download. This contains the IP address of the peer that hosts the file, the port number of that peer and the connection speed of the peer.
Browse a user's files	Napster peer sends this message to the server for requesting to browser the shared file of a specific nickname.
Browse response	Napster server sends this message to the peer to show the details of the shared files of a specific nickname.
Alternate download request	Napster peer will send this message to the server when a firewalled peer hosts the request file.

The server will respond with a "Download ack" message, which contains information that is more detailed. If the "Download ack" message says that the remote client's data port is zero, the requesting peer must send an "Alternate download request" message to the server to request the remote peer send the data to it. In this case, it waits for the remote peer to connect to its own data port open for Napster sharing. In the case that the sharing peer is not behind a firewall, the requesting peer makes a transmission control protocol (TCP) [8] connection to the data port specified in the "Download ack" messages from the server. The remote peer should accept the connection and immediately send one ASCII character '1' (ASCII 29). Once the downloading peer reads this char, it will send a request for the file it wish to download - first sending the string "GET" in a single packet a, then sending <mynick> "<filename>" <offset>. The <mynick> is your Napster user name, the <filename> is the file you wish to download, and the <offset> is the byte offset of the file to begin with. The remote peer will then return the file size, or an error message such as "INVALID REQUEST" or "FILE NOT SHARED". Immediately following the file size is where the data stream for the file begins. Figure 4 shows the logical flow of the Napster messages between the Napster peer and Napster Server, and Table 4 summarizes the messages used in the Napster protocol [7].

4. INTEROPERABILITY OF PROTOCOLS

The purpose of P2P file sharing is to provide an efficient method for sharing (meaning search and download) data among peers, therefore interoperation among different P2P file sharing applications is essential. Unfortunately, the three popular file sharing

applications are not interoperable because their underlying protocols are different. In this paper, we propose a P2P gateway, which works independently of the file sharing applications, to receive messages from all different protocols and convert them into the specific protocol, which the remote file sharing application attaches. Moreover, all outgoing messages will be converted into different formats for each protocol and then sent out to the network. We illustrate the framework for implementing the P2P gateway in Figure 6. The gateway acts as a middleware to handle all incoming and outgoing messages of the P2P file sharing application. In Table 5, we group the messages of each protocol into different classes according to their functionalities. The P2P gateway performs the conversion, for instance converting the "HandshakeRequest" message of Freenet to the "Ping" message of Gnutella and the "Login" massage of Napster. Besides, the gateway can keep a dynamic database that stores the information of the current connected host. This database can provide the IP address of the Gnutella and Freenet peers therefore enable the Napster browser. The gateway can also cache information about the files on the Freenet and the Gnutella peers by storing the hash file index and a signature file that are used to identify the peer. The signature file can be a MD5 digest of the file index. Therefore, if file in the shared folder is alerted, the digest will be altered. Security is an important issue for P2P file sharing network, an example will be the DOS due to query reply messages with the IP addresses of the web server. To avoid this attack, the protocol can request an additional peer registry & discovery message, before the http request. If the IP address in the query reply message does not point to a P2P file sharing application, the P2P file sharing application will not get an appropriate response, and therefore terminates the http request.

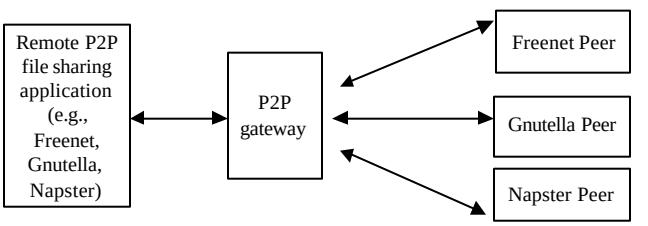

Figure 6. The framework for P2P files sharing protocols integration.

Table 5. Comparison of Peer registry, query and file download.

	Freenet	**Gnutella**	**Napster**
Peer registry & discovery	HandshakeRequest HandshakeReplay	Ping Pong	Login Login ack
Query	ClientGet DataFound	Query QueryHit	Client search request Search response Browse a user's file Browse response Download request Download ack
Data download protocol	Freenet DataChunk	HTTP	TCP
Data transfer mechanism for firewalled peer	Not available	Push	Alternate download request

5. CONCLUSIONS

In this paper, we conducted a comprehensive analysis of P2P file sharing applications, including Freenet, Gnutella, and Napster. An in-depth exploration of their protocols and architectures is illustrated and summarized. Our studies focused on the message exchanges between peers and servers under different P2P file sharing schemes and models. In order to maximize the benefit of P2P file sharing application, we propose a framework to integrate various P2P file sharing protocols by using a P2P gateway. The P2P gateway acts as a message converter, and operates in between peers. It converts incoming and outgoing messages to an understandable format for the requesting and responding peers. Interoperability of these file sharing protocols will facilitate file sharing. It is also extensible to the application to knowledge management system, in which users share and exchange knowledge, documents in organizations. Furthermore, the interoperability of these file protocols may enhance the existing Internet search mechanism, which only searches against the keywords or metadata of the webpage host in web servers.

ACKNOWLEDGMENTS

This research is supported in part by the Innovation and Technology Fund of Hong Kong (AF/168/99), and the Hong Kong Research Grants Council through a Direct Allocation Grant (DAG00/01.BM35 and DAG01/02.BM16).

REFERENCES

1. Freenet, http://freenetproject.org/lang/en/.
2. Freenet documentation, http://freenet.sourceforge.net/doc/book.html.
3. Gnutella, http://www.gnutelliums.com
4. Gnutella Protocol Specification Version 0.4, http://www.clip2.com
5. I. Clarke. (1999). "A distributed decentralized information storage and retrieval system," unpublished report. Division of Informatics, University of Edinburgh, http://www.freenetproject.org.
6. Napster Inc., http://www.napster.com.
7. "Napster protocol specification," http://opennap. sourceforge.net/napster.txt.
8. Postel, J. (1981). Transmission Control Protocol, RFC-793. Network Information Center, SRI International.
9. R.Fuekding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. (1999). Hypertext transfer protocol – http/1.1 Technical Report RFC2626, IETF.
10. What is peer-to-peer?. Peer-to-Peer working group. http://www.p2pwg.org.
11. S. Saroiu, P. Gummadi, and S. Gribbe (2002). "A measurement study of peer-to-peer file sharing systems". *Technical report UW-CSE-01-06002*, University of Washington.